Handling congestion control for Internet of Thing application

Synopsis of the PhD Thesis

Submitted by

Sneha R. Deshmukh

(Roll No P007)

Department of Information Technology

Supervisor

Prof. Vijay T. Raisinghani

submitted in partial fulfilment of the requirements of

DOCTOR OF PHILOSOPHY (Engineering)

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING
SVKM's NARSEE MONJEE INSTITUTE OF MANAGEMENT STUDIES (Declared as Deemed-to-be University Under Section 3 of the UGC Act, 1956)
V. L. Mehta Road, Vile Parle (West)
MUMBAI –400056

November 2022

TABLE OF CONTENTS

ABSTRACT 5
1. INTRODUCTION 6
2. OBJECTIVES 8
3. LITERATURE REVIEW 9
3.1 Constrained Application Protocol (CoAP) 9
3.2 RTO-based protocols 10
3.3 Queue monitoring protocols 10
3.4 Rate-based protocols 13
3.5 Backoff based protocols 13
3.6 Artificial intelligence protocols 13
3.7 Motivation 13
3.8 Problem Definition 14
3.9 Contributions 15
4. CONTRIBUTION (Proposed protocols) 17
4.1 Adaptive Congestion Control Algorithm (AdCoCoA) 17
4.2 Adaptive Congestion Control Algorithm with backoff (ACCB) 18
4.3 Performance Evaluation 19
4.3.1 Simulation setup 19
4.3.2 Results 21
4.4 Discussion 26
5. CONCLUSION 28
6. FUTURE SCOPE 29
7. PUBLICATIONS 30
REFERENCES 31

List of Figures

Figure 1: CoAP message types 9
Figure 2: Working Binary exponential backoff (BEB) in CoAP 9
Figure 3: Connected cars scenarios 14
Figure 4: An overview of the AdCoCoA mechanism 17
Figure 5: An overview of the ACCB mechanism 18
Figure 6: Topologies (a) Linear (b) Random 21
Figure 7: *Goodput v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes* 22
Figure 8: *Packet delivery ratio v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes* 23
Figure 9: *Retransmission ratio v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes* 23
Figure 10: *Packet sending rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes* 23
Figure 11: *Goodput v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes* 25
Figure 12: *Packet delivery ratio v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes* 25
Figure 13: *Retransmission ratio v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes* 26
Figure 14: *Packet sending rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes* 26

List of tables

Table 1: Summary of congestion control protocols 13
Table 2: Summary of contribution 16
Table 3: Simulation parameters 20

ABSTRACT

The Internet of things (IoT) is a system of interconnected computing devices, mechanical and digital devices, objects with unique identifiers, and able to communicate over a network without any intervention. The Internet Engineering Task Force (IETF) has standardized the CoCoA+ protocol for CoAP. CoCoA+ derived from LinuxRTO performs better than basic CoAP in many scenarios. CoAP performs significantly worse in light traffic and bursty traffic environment. The CoAP standard has an exponential backoff congestion control mechanism. This backoff mechanism may not be adequate for all IoT applications. The characteristics of each IoT application would be different. Congestion control is one of the critical challenges in IoT networks. Further, the events such as unnecessary retransmissions and packet collision caused due to links with high losses and packet transmission errors may lead to network congestion. Various congestion-handling algorithms for CoAP have been defined to enrich the performance of IoT applications. We propose and design two protocols to handle congestion in CoAP networks. Our first protocol an adaptive congestion control protocol for CoAP derived from CoCoA+, named Adaptive Congestion Control Algorithm (AdCoCoA). The proposed algorithm estimates RTO using dynamic factors instead of fixed values. This dynamic adaptation helps to improve CoAP performance and reduce retransmissions. The results show AdCoCoA has lower RTO values, retransmission index (11.6%); higher goodput (71.3%), packet delivery ratio (36.4%), and transmission rate (55.6%) compared with CoCoA+. Our second protocol is Adaptive Congestion Control with a Backoff algorithm (ACCB). AACB is an extension to our earlier protocol AdCoCoA. The proposed algorithm estimates RTT, RTTVAR, and RTO using dynamic factors instead of fixed values. Also, the backoff mechanism has dynamic factors to estimate the RTO value on retransmissions. This dynamic adaptation helps to improve CoAP performance and reduce retransmissions. The results show ACCB has significantly higher goodput (49.5%, 436.5%, 312.7%), packet delivery ratio (10.1%, 56%, 23.3%), and transmission rate (37.7%, 265%, 175.3%); compared with CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. The results show ACCB has significantly higher goodput (60.5%, 482%,202.1%), packet delivery ratio (7.6%, 60.6%, 26%), and transmission rate (40.9%, 284%, 146.45%); compared with CoAP, CoCoA+ and AdCoCoA respectively in random walk scenario. ACCB has similar retransmission ratio compared with CoAP, CoCoA+ and AdCoCoA respectively in both the scenarios.

1. INTRODUCTION

The Internet of things (IoT) has a significant role in today's applications with the fifth generation of cellular technology. IoT networks are a network of things capable of sending, receiving, or exchanging information. The IoT devices have limited battery, memory, and processing capabilities. They can communicate over links that may have higher packet error rates and low throughput. These characteristics may lead to more retransmissions, which eventually may cause congestion in the network. Either loss of packet or acknowledgment (ACK) is commonly considered an indication of congestion. In IoT networks, packet or ACK loss can happen due to (i) lossy links, link errors, or weak signal strength and (ii) delay in receiving the ACK, which leads to spurious retransmissions. Hence, an efficient congestion control mechanism is required to minimize unnecessary retransmissions and accurately estimate the retransmission time out (RTO) value. The IETF has defined a protocol stack for these IoT devices. In the IoT protocol stack, Constrained Application Protocol (CoAP) operates over UDP as the transport protocol – supports lightweight internet applications.

CoAP [1] uses a request/response interactive model between the application endpoints. A message identifier field helps detect duplicate messages. The Confirmable (CON) message requires an acknowledgment (ACK), whereas a Non-confirmable (NON) message doesn't require an ACK. A Reset (RST) message indicates to the sender that the receiver is unable to process the message. CoAP provides optional reliability using the RTO mechanism and an exponential backoff policy. It chooses the initial RTO value randomly, and later it uses a binary exponential backoff (BEB) mechanism for computing the RTO value. The random initial value could lead to unnecessary retransmissions and, thus, network congestion [2]. Also, the CoAP backoff mechanism often fails to utilize the network dynamics to the best of its traffic conditions. Currently, the CoAP Simple Congestion Control/Advanced (CoCoA) [2] is the standard protocol for CoAP by IETF. CoCoA considers Round Trip Time (RTT) for estimating Retransmission Timeout (RTO) along with a Variable Backoff Factor (VBF) and aging mechanisms to provide flexibility and monitored adaptation suited to the dynamic nature of IoT networks. Each IoT application has different characteristics and behavior, hence a single standard algorithm may be inadequate to handle the congestion in the network.

Recent studies [34], [35] show that CoCoA+ can perform better than basic CoAP in many scenarios but performs significantly poorly in bursty traffic environments in a large-scale network. In [7], [36], the authors have highlighted the drawbacks of CoCoA+. Many researchers have proposed different mechanisms to overcome the weaknesses of CoCoA+. Some researchers aim to improve the performance of CoAP in a bursty traffic environment by measuring the RTT precisely and monitoring network conditions to minimize spurious retransmission [7]. Some researchers use a rate-based mechanism [18], which utilizes the maximum bandwidth of the bottleneck link and ensures fairness. Existing protocols use fixed scaling factors for computations of the RTO, which might not be suitable for a wide range of IoT scenarios, and thus lead to higher retransmissions. Dynamic scaling factors could estimate RTO to be large enough to minimize retransmissions.

We propose a dynamic scaling factor for estimating RTT, RTTVAR, and RTO to overcome the fixed scaling values. We have proposed multiple protocols which have been listed in our contributions section 3.9. In this synopsis, we have discussed two of the protocols in detail. The rest are presented in the thesis.

Our first protocol, AdCoCoA, estimates RTO using dynamic scaling factors for smoothed *round trip time* (RTT), RTT variation (RTTVAR), and RTO instead of fixed values. The dynamic scaling factors are estimated using a single network status parameter, i.e., RTT measurement monitoring. Our second protocol, ACCB, estimates RTO using dynamic scaling factors for smoothed round trip time (RTT), RTT variation (RTTVAR), and RTO. The dynamic scaling factors are estimated by considering multiple network status parameters. Also, ACCB has a dynamic backoff mechanism for retransmission. The proposed models (AdCoCoA and ACCB) are evaluated against CoAP,and CoCoA+.

2. OBJECTIVES

Existing protocols use fixed scaling factors for computations of the RTO, which might not be suitable for a wide range of IoT scenarios, and thus lead to higher retransmissions.

To define a congestion control protocol for Internet of Things (IoT) application (connected cars) with the following characteristics –

- The protocol should minimize unnecessary retransmissions and optimize the RTT and RTO.
- The protocol should handle variations in mobility speed, data traffic type, number of senders, density of nodes, interference level, and packet size.

3. LITERATURE REVIEW

3.1 Constrained Application Protocol (CoAP)

CoAP (RFC 7252) [1] has a client-server type of model with requests and responses. CoAP uses a stop-and-wait approach, retransmissions, and exponential backoff for confirmable (CON) messages. Figure 1(sourced from reference [1]) shows four types of CoAP messages: Reset (RST), Acknowledgement (ACK), Non-confirmable (NON), and Confirmable (CON). A message identifier helps in duplicate identification. An ACK is returned for a CON message. ACK is not required for NON. RST from the receiver to the sender indicates that the receiver could not process the message. CoAP implements an optional reliability mechanism. When the ACK is not received for a transmitted CON, the sender retransmits the message.

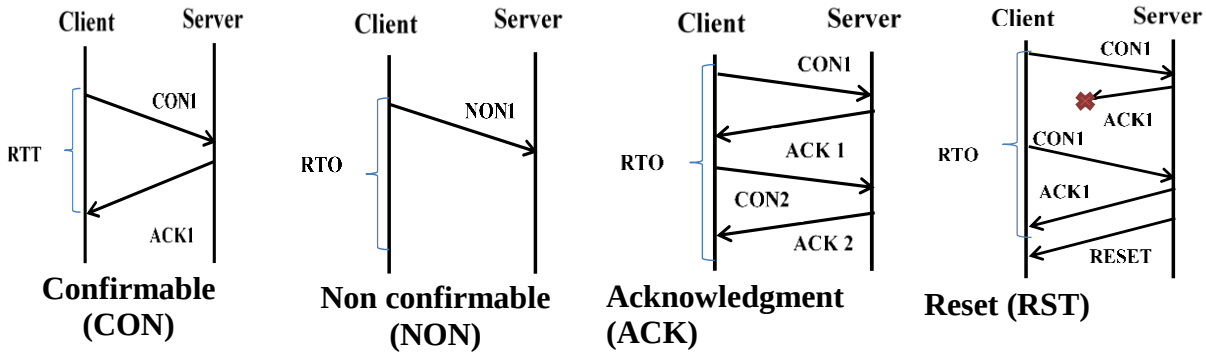


Figure 1: CoAP message types

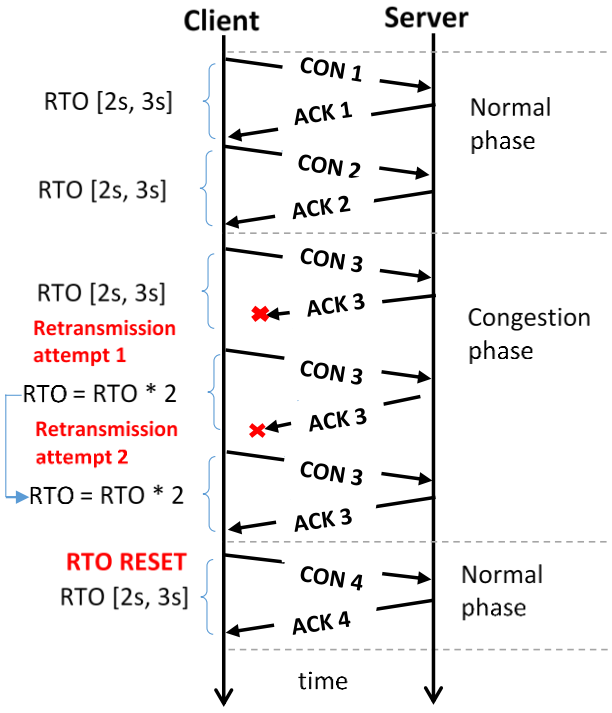


Figure 2: Working Binary exponential backoff (BEB) in CoAP

On sending a CON message, the sender chooses an initial value from 2s to 3s for the RTO. The sender assumes the CON message is lost. If an ACK is not received from the destination before the expiry of the RTO timer. The sender retransmits and doubles the current RTO value to reduce needless retransmissions to avoid congestion, as shown in figure 2 (sourced from reference [1]). The

sender does not consider RTT to compute the RTO. As a result, the RTO value would tend to be inaccurate and could lead to unnecessary retransmissions. The RTO value will grow exponentially if the retransmissions increase, leading to an increase in the retransmission delay.

We broadly classify the congestion control mechanisms for CoAP networks into five categories – RTO-based, queue-monitoring, rate-based, backoff-based, and artificial intelligence (AI). Table 1 shows the summary of these protocols.

3.2 RTO-based protocols

RTO-based mechanisms: CoCoA [2] calculates RTO by using the value of Round trip time (RTT). CoCoA+ [3] has step-function techniques with scaling factors to calculate the RTO for retransmission. CoCoA-E [4] is a modified version of CoCoA+, which uses a new scaling factor based on the Eifel retransmission timer. Bhalerao et al. [5] have further improved the RTO estimation mechanism of CoCoA+ using an adaptive technique based on retransmission attempts. Lee et al. [6] have defined a method for congestion control that uses RTT. This mechanism considers retransmission count for accurately measuring the RTT value and estimating the RTO value. pCoCoA [7] estimates the RTT precisely using timestamps and a modified RTO estimation mechanism. Hence, it helps minimize unnecessary retransmissions.

An improved adaptive CoAP [8] determines the RTO value using packet loss ratio and RTT. RTT-CoAP [9] has a novel approach to detecting congestion in the network. It checks the variance of RTT against predefined thresholds for loss of COAP messages. CoAP Eifel [10] is a modified version of CoAP which uses only strong RTT to estimate the RTO value. In AdCoCoA [11], link quality, delay, and RTT deviation are considered to calculate the RTO value. CACC [12] categorizes RTT into three types, as per the receipt of an ACK – without retransmission (strong) or with retransmission (weak; one or more retransmissions) or no ACK at all (failed), to determine the condition of the network, and adapt the congestion control mechanism. FASOR [13] determines whether the loss of packets is because of poor links or congestion in the network. FASOR has three distinctive features – an adaptation of the RTO backoff and RTO computation which can be fast or slow. DCC-CoAP [14] efficiently predicts network congestion; it combines the separation of nodes and multiple RTT measurements, reducing CoAP message loss. CoCoA++ [15] uses the delay gradients method to measure network congestion and introduces a new backoff mechanism called *probabilistic backoff* to deal with congestion.

3.3 Queue monitoring protocols

Queue monitoring mechanisms: In a Bird flock model [16], a node monitors its neighbor node's buffer at each hop to avoid congestion. The backpressure congestion-control mechanism [17] uses the MAC and network layer's information to prevent congestion at each hop. Rate-based mechanism: CoAP-R [18] is designed to utilize the maximum available bandwidth and uses max-min fairness to allocate bandwidth. BDP-CoAP [19] design copes with lossy links and problems of unfair access over the short term, which typically occurs in IoT networks.

Protocol	Backoff mechanism	Derived from	Strength	Weakness
CoAP std in (2014) [1]	Binary exponential backoff	None	Simple mechanism	1. RTT not considered for computing RTO 2. Lack of RTO aging mechanism
Fullbackoff variants (2018) [37]	*Fullbackoff1* and F*ullbackoff2*	BEB	Simple mechanism	*Fullbackoff1* decreases RTO value drastically on completing successful transmission
GST (2020) [38]	GST backoff	*fullbackoff2*	Decreases RTO value gradually instead of drastically	Maintains the history of RTO values
CoCoA (2014) [2]	Binary exponential backoff	Linux-RTO	1.Considers RTT to compute RTO 2.Indirectly considers link quality by strong /weak RTT measures	1. Equal weights for strong/weak RTT 2. RTO history not considered for reinitiated connections; Lack of RTO aging mechanism 3. Excessive RTO growth (doubling on every retransmission) and
CoCoA+ (2015) [3]	Variable backoff factor	CoCoA	1. Gradual increase in RTO on every retransmission 2. RTO aging	1. Step function intervals and factors are fixed 2. Due to sudden increase in $RTT_{measured}$ the RTT and RTO values could be close
CoCoA-E (2013) [4]	Variable backoff factor	CoCoA and Eifel	1. Dynamic scaling factor used for RTT-RTO estimation 2. RTO aging	Unable to adapt the RTO value if there is a sudden increase in packet sending rate
4-state Strong (2016) [5]	4-state Variable backoff factor	CoCoA	Flexible backoff values for four level retransmission attempts	RTO aging mechanism – fixed interval
Enhanced CoCoA (2016) [6]	Variable backoff factor	CoCoA	Determines RTT value accurately using retransmission count parameter	1. Fixed weights for computing RTT and RTO 2. RTO aging mechanism – fixed interval
pCoCoA (2018) [7]	Variable backoff factor	CoCoA	1. Estimates RTT accurately which helps to compute accurate RTO value 2. Adjusts RTO if spurious retransmission detected	1.Spurious retransmission possibility due to sudden increase in traffic 2. Fixed interval for RTO aging mechanism 3. High computation mechanism
RTT CoAP (2018) [9]	Rate control (AIAD)	CoAP and RTT monitoring	Determines RTT variance growth	1.Considers single factor (RTT variance) to determine network status 2. High computation mechanism
CoCoA++ (2019) [15]	Probabilistic Backoff Factor	CoCoA	Considers the RTT variance growth to determine recent network conditions	1.Considers single factor (RTT variance) to determine network status 2.RTT monitoring duration may not be suitable for some applications 3. High computation mechanism
FASOR (2019) [13]	Retransmission timer backoff	CoCoA	1.Computes Fast and Slow RTO 2.Self-adapting backoff mechanism	Considers single factor (ACK) to determine network status
Improved Adaptive CoAP	Dynamic RTO	CoAP	1.Computes packet loss ratio *(PLR)* which indicates network status 2.Adaptive scaling factors are used for RTO estimation	1.Considers single factor (packet loss ratio) to determine scaling factor used to estimate RTO value 2.Overhead of computing packet loss ratio per RTT measurement

(2019) [8]				
CACC (2020) [12]	Dynamic RTO	CoCoA	1. Considers strong, weak and failed RTT measurements 2.Adaptive scaling factors are used for RTO estimation	1. Considers single factor (RTT measurements) to determine scaling factor used to estimate RTO value 2. High computation mechanism
CoAP-Eifel (2020) [10]	Binary exponential backoff	Eifel retransmission	Considers strong RTT	1. Excessive RTO growth (on retransmission) 2. Lack of RTO aging
AdCoCoA (2020) [11]	Variable backoff factor	CoCoA	Considers RTT deviations for estimating dynamic scaling factors	Considers single factor (RTT variance) to determine scaling factor used to estimate RTO value
DCC-CoAP (2020) [14]	Variable backoff factor	CoCoA	Determines signal strength to estimate dynamic scaling factor	Considers single factor (signal strength) to determine scaling factor used to estimate RTO value
Genetic CoCoA+++ (2020) [39]	Probabilistic Backoff Factor	CoCoA++	Considers RTT_{min} to estimate delay gradient appropriately	1.Considers single factor (RTT variance) to determine network status 2.RTT monitoring duration may not be suitable for some applications
Back Pressure congestion control mechanism (2013) [17]	Rate based	Back pressure mechanism	1. Prevents congestion by queue monitoring 2.Congestion notification	1. Higher end-to-end delay due to hop-by-hop congestion determining mechanism 2. Fixed threshold value for queue length 3. Overhead of control messages
Bird Flocking (2015) [16]	Rate based	Bird flock concept	No overhead of control messages	1. Energy consumption due to monitoring and computations 2. Routing table storage overhead 3. Higher end-to-end delay due to hop-by-hop congestion determining mechanism
CoAP-R (2018) [18]	Rate control	CoAP and RTT monitoring	Determines link capacity	Considers single factor (node's link capacity) to determine network status
BDP-CoAP (2019) [19]	Rate control	TCP BBR	Determines the bandwidth of bottleneck link in the network	Considers single factor (bandwidth delay product of link) to determine network status
mlCoCoA (2018) [40]	Variable backoff factor	CoCoA and Machine learning	Considers load in network, packet size and physical layer PDR to determine values of scaling factors	1. Optimal values limited 2. High computational mechanism
CCCLA (2019) [41]	Rate control	Learning automata	Considers multiple parameters to determine network status	1. May not be able to define optimal network parameters in dynamic scenarios 2. High computation mechanism

FLCoCoA(2021) [42]	Adaptive-boundary backoff factor (ABF)	CoCoA	Considers multidimensional factors to determine RTO value and backoff boundary	High computational mechanism
RCOAP (2021) [43]	Variable backoff factor	CoCoA	Determines the bandwidth of bottleneck link in the network to estimate initial transmission rate	High computational mechanism

Table 1: Summary of congestion control protocols

3.4 Rate-based protocols

Rate-based mechanisms: CoAP-R [18] is designed to achieve maximum bottleneck link bandwidth and allocates the network resources based on max-min fairness. BDP-CoAP [19] is designed to cope with lossy links and the short-term unfairness of channel access that is typically observed in IoT networks. RCOAP [20] determines the available bandwidth and probes the sender to estimate the transmission rate.

3.5 Backoff based protocols

Backoff based mechanisms: In [37], full backoff has two variants – fullbackoff1 and fullbackoff2. Fullbackoff1 resets RTO value drastically, whereas fullbackoff2 resets RTO value gradually. In [38], the Geometric Sequence Technique (GST) uses the history of RTO values and the number of retransmissions for RTO estimation in CoAP.

3.6 Artificial intelligence protocols

Artificial intelligence (AI) based mechanisms: mlCoCoA [40] is enhanced CoCoA which uses a machine learning mechanism to tune the RTO parameter values using node count, packet size, and packet delivery ratio (PDR). In CCCLA [41], a group of tunable parameters, is used to control congestion in the network and enhance the network performance using learning automata. FLCoCoA [43] used a multidimensional congestion estimator to predict congestion conditions in the network and proposed a new congestion-aware backoff strategy based on an adaptive-boundary backoff factor evaluated using a fuzzy logic system (FLS).

3.7 Motivation

In connected cars application, the cars determine the events and inform traffic jams, sudden weather changes, accidents, and so on to the server[45] [46] . The majority of these application require multi-hop forwarding of the packets as shown in figure 2. These updating packets and forwarding cause congestion at the server.

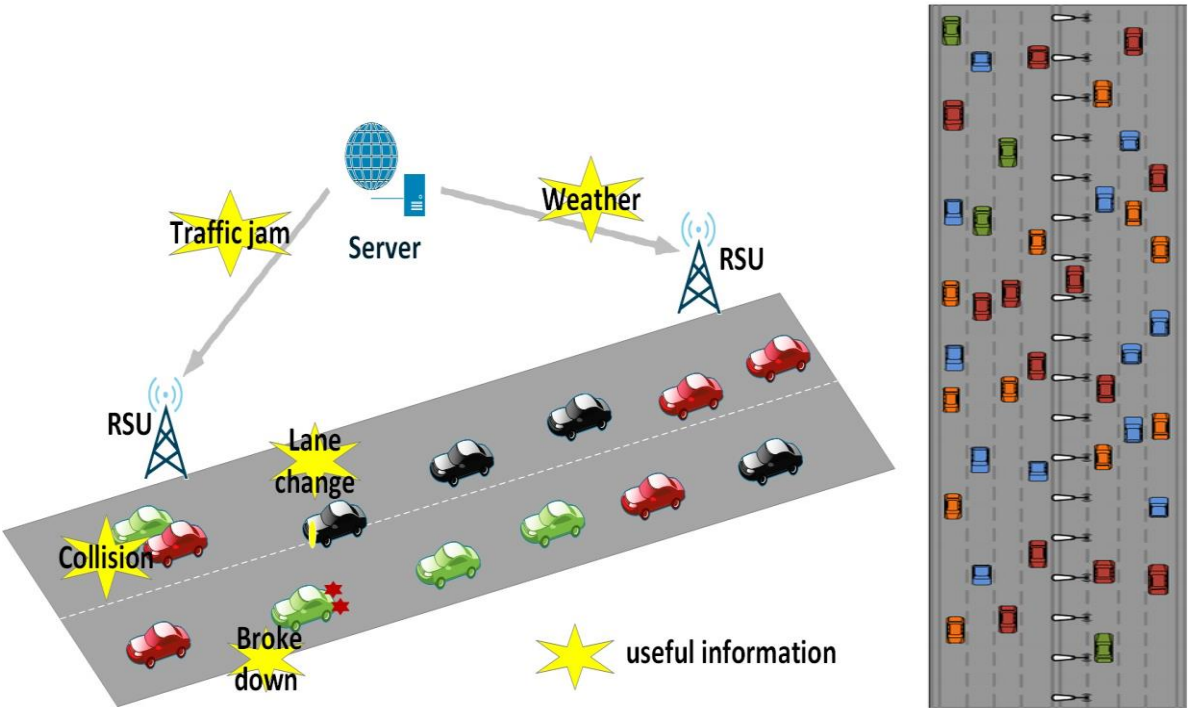


Figure 3: Connected cars scenarios

Based on the literature review, most existing algorithms use fixed scaling factors for estimating smoothed RTT, RTTVAR, and RTO. Few algorithms are designed only to handle either light or bursty data traffic. The fixed scaling factors will be inadequate for many IoT applications. Using fixed scaling factors may lead to unnecessary retransmission due to the inability to adapt to sudden change in RTT measurements (increase or decrease). Hence, there is a strong need to develop an algorithm that considers network status, to estimate RTO value.

3.8 Problem Definition

Our aim is to propose a protocol to handle congestion control for Internet of things (IoT) application (connected cars). The protocol should have the following characteristics:

- Minimizes unnecessary retransmissions

  The algorithm should estimate the appropriate (optimized difference) between RTT and RTO

- Adaptive to changes in network conditions (link quality or delay)

  The proposed protocol should monitor changes in network status parameters to get network insights. The network status parameter like change in RTT measurements, packet loss ratio, retransmission ratio, etc.

- Handles variation in the number of senders, mobility speed, interference level, density of nodes, data traffic, and packet size

  Based on the network status, dynamic scaling factors should be estimated. The dynamic scaling factors will be used to calculate smoothed RTT, RTT variance, and RTO.

3.9 Contributions

We have listed below the various protocols we have defined. Table 2 below summarizes our contributions.

Protocol 1: Adaptive Congestion Control Algorithm (AdCoCoA)

RTT, RTTVAR, and RTO estimation use dynamic scaling factors instead of fixed scaling factors

- Single network parameter (RTT deviation) used to estimate dynamic scaling factors

Protocol 2: Enhanced Adaptive Congestion Control Algorithm (AdCoCoA+)

The RTT, RTTVAR, and RTO estimation use dynamic scaling factors similar to AdCoCoA. The algorithm has improved backoff mechanism which use Adaptive scaling factor (ASF) to scale up the RTO on retransmission.

Protocol 3: Adaptive Congestion Control with Variable backoff (ACCVB)

- RTT and RTTVAR estimation use dynamic scaling factors (multiple network parameters). Backoff mechanism is the same as AdCoCoA

Protocol 4: Adaptive Congestion Control with backoff variants

- Designed multiple backoff algorithm variants for ACCVB. Backoff variations by additive or multiplicative approach, considering either single or combinations of multiple parameters (RTT deviation, packet loss ratio, retransmission ratio) to estimate the dynamic scaling factors for RTO on retransmissions

Protocol 5: Adaptive Congestion Control Algorithm with backoff (ACCB)

- RTT and RTTVAR estimation use dynamic scaling factors (multiple network parameters). Improved RTO estimation formula.
- To estimate dynamic scaling factors, consider RTT deviation, packet loss ratio, and retransmission ratio.
- We have introduced a new backoff mechanism, Adaptive backoff dynamic scaling (ABDS) to compute RTO value using a dynamic scaling factor on retransmission attempts.

SR No	Protocol	RTT, RTTVAR, RTO factors	Parameters measured for RTT, RTTVAR and RTO factors	Backoff factor(s)	Parameters measured for backoff factor(s)	Type of contribution
1.	AdCoCoA	Dynamic scaling factors	RTT deviation	Variable backoff factor (VBF)	RTO	Major
2.	AdCoCoA+	Dynamic scaling factors	RTT deviation	Adaptive scaling factors (ASF)	RTO difference	Minor
3.	ACCVB	Dynamic scaling factors	RTT deviation, retransmission ratio, packet loss ratio	Variable backoff factor (VBF)	RTO	Minor
4.	ACC with adaptive backoff variations	Dynamic scaling factors	RTT deviation, retransmission ratio, packet loss ratio	10 variants of backoff mechanism	Either single parameter or combinations of parameters (RTT deviation, retransmission ratio, packet loss ratio)	Major
5.	ACCB	Dynamic scaling factors	RTT deviation, retransmission ratio, packet loss ratio	Adaptive backoff with dynamic scaling (ABDS)	RTT deviation	Major

Table 2: Summary of contribution

4. CONTRIBUTION (Proposed protocols)

As discussed in the literature review, most existing congestion control protocols for CoAP use fixed scaling factors for the RTO estimator. These fixed factors might not be suitable in many IoT scenarios, for example, connected cars or self-driven cars. Hence, dynamic scaling factors may help to improve the performance of CoAP for such applications. Our algorithms consider varying density (low to high) and mobility speed of vehicles (slow to fast) and adapts as per the characteristic of the application. In this synopsis, we have presented two of our algorithms, the rest are discussed in the thesis.

4.1 Adaptive Congestion Control Algorithm (AdCoCoA)

AdCoCoA is an enhancement of CoCoA+, which estimates RTT, RTTVAR, and RTO using dynamic scaling factors rather than fixed factors. AdCoCoA eliminates the need to maintain two RTO estimates that are strong and weak. It measures RTT using a timestamp, hence only strong RTT contributes to RTO estimation. It determines dynamic scaling and smoothing factors for estimating RTT, RTTVAR, and RTO based on the network status, rather than fixed values. The variable backoff factor (VBF) mechanism borrowed from CoCoA+ is used for RTO estimation on retransmission attempts. Figure 4 shows the overview of the AdCoCoA mechanism for CoAP.

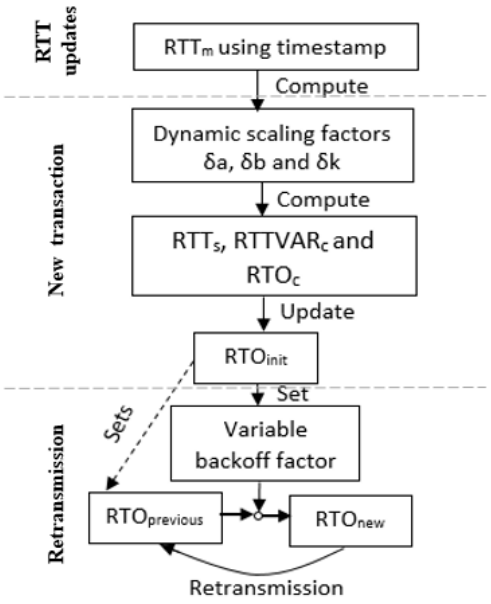


Figure 4: An overview of the AdCoCoA mechanism

To calculate *RTT deviation (delta)*.

$$delta = ((RTT_m - RTT_{current})/ RTT_{current}) \quad \text{(eq 1)}$$

To calculate dynamic smoothing factors δ_a, δ_b, and δ_K

δ_a = delta * alpha;
δ_b = delta * beta;
δ_K = delta * K; (eq 2)

To calculate smoothed RTT, RTTVAR, and RTO values, following equations are used:

$RTT_{current} = (1 - \delta_a) * RTT_m + \delta_a * RTT_{current}$ (eq 3)

$RTTVAR_{current} = (1 - \delta_b) * RTTVAR_{previous} + \delta_b * | RTT_m - RTT_{current}|$ (eq 4)

$RTO_{current} = RTT_{current} + \delta_K * RTTVAR_{current}$ (eq 5)

$RTO_{overall} = 0.5*RTO_{previous} + 0.5 * RTO_{current}$ (eq 6)

The Variable backoff factor (VBF) mechanism calculates the RTO value on retransmissions like CoCoA+. RTO = $RTO_{previous}$ * VBF. If the $RTO_{previous}$ value < 1 second, VBF =3. If the $RTO_{previous}$ value is between 1 to 3 seconds, VBF=2. If the $RTO_{previous}$ value > 3 seconds, VBF = 1.3.

4.2 Adaptive Congestion Control Algorithm with backoff (ACCB)

ACCB is an enhancement of AdCoCoA, which (i) estimates dynamic scaling factors using multiple network parameters instead of a single parameter and (ii) uses an Adaptive backoff (ABF) mechanism instead of the variable backoff (VBF) mechanism. AdCoCoA determines dynamic scaling factor using only RTT variance, whereas ACCB determines using RTT variance, retransmission ratio, and packet loss ratio.

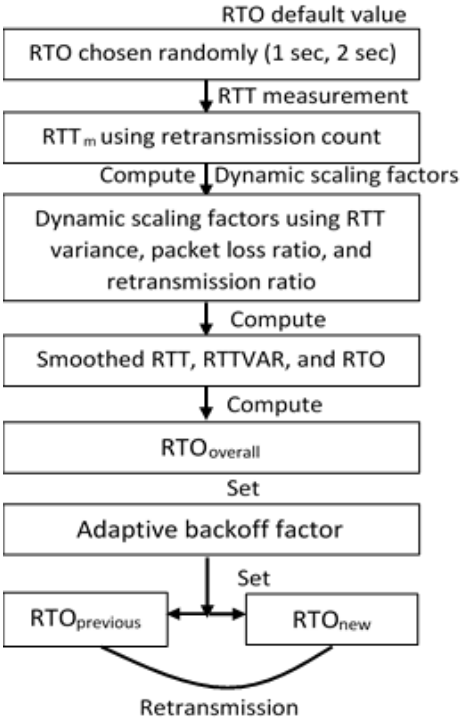


Figure 5: An overview of the ACCB mechanism

Figure 5 shows the overview of the ACCB mechanism for CoAP. ACCB measures RTT using a timestamp; hence only strong RTT contributes to RTO estimation. On sending the CON message, if the receiver is unknown, then the default CoAP mechanism is used for the first time. On estimating the first measured RTT, ACCB calculates Smoothed RTT, RTTVAR, and RTO, the same as AdCoCoA. ACCB uses dynamic scaling and smoothing factors for subsequent measured RTTs to calculate RTT, RTTVAR, and RTO values. ACCB calculates dynamic scaling and smoothing characteristics for estimating RTT, RTTVAR, and RTO.

To calculate *RTT deviation, retransmission ratio,* and p*acket loss ratio,* following equations are used:

RTT deviation = $((RTT_m - RTT_{current})/ RTT_{current})$ (eq 7)

If the *RTT deviation* exceeds one, the *RTT deviation* value equals 0.9.

$$retransmission\ ratio\ = \frac{\text{number of packets retransmitted}}{number\ of\ packets\ transmitted} \quad \text{(eq 8)}$$

If the *retransmission ratio* is zero, the *retransmission ratio* value equals 0.1.

$$packet\ loss\ ratio\ = \frac{\text{number of packets loss}}{number\ of\ packets\ transmitted} \quad \text{(eq 9)}$$

If the *packet loss ratio* is zero, then the *packet loss ratio* value equals 0.1.

To calculate dynamic smoothing factors *alpha* and *beta.*

alpha = ((*RTT deviation* * 0.5) + (*retransmission ratio* * 0.25) + (*packet loss ratio* * 0.25))

beta = alpha * 0.5 ; (eq 10)

To calculate smoothed RTT, RTTVAR, and RTO values.

$RTT_{current} = (1 - alpha) * RTT_m + alpha * RTT_{current}$ (eq 11)

$RTTVAR_{current} = (1 - beta) * RTTVAR_{previous} + beta * | RTT_m - RTT_{current}|$ (eq 12)

$RTO_{current} = RTT_{current} + K * RTTVAR_{current}$ (eq 13)

Where K is the scaling factor. If the *RTT deviation* value exceeds one, K =6; else, K=4.

$RTO_{overall} = 0.5*RTO_{previous} + 0.5 * RTO_{current}$ (eq 14)

The Adaptive backoff dynamic scaling (ABDS) mechanism calculates the RTO value on retransmissions. If $RTO_{overall}$ is less than one second, $RTO_{new} = RTO_{previous} + (1 + RTT\ deviation)$*; else,* $RTO_{new} = RTO_{previous} + RTT\ deviation.$

In this section, we described the design and working of the ACCB algorithm. The following section discusses the simulations used to validate ACCB against CoAP, CoCoA+, and AdCoCoA.

4.3 Performance Evaluation

In this section, we compare the performance of ACCB with CoAP, CoCoA+, and AdCoCoA through simulations. This section includes the simulator's configuration, the traffic scenario, the network topologies, and the performance metrics used to carry out the performance evaluations.

4.3.1 Simulation setup

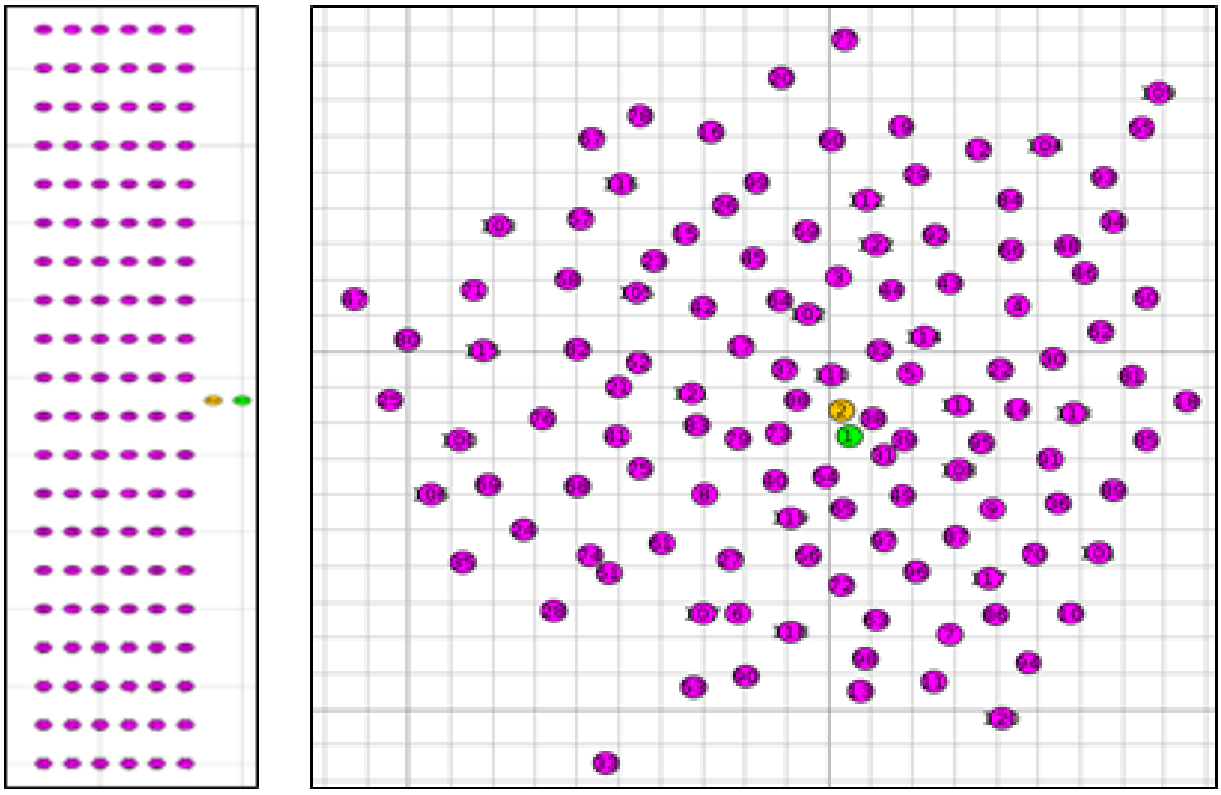
We use the Cooja simulator [20], a simulation platform included in Contiki 3.0 toolset [20], that allows emulating off-the-shelf wireless sensor node hardware. At the physical (PHY) and MAC layers, the nodes implement IEEE 802.15.4, using a transmission rate of 250 kbps in the 2.4 GHz radio band. The Cooja simulator and Zolerita Z1 motes are used for clients and servers. Z1 motes have 92KB of ROM, enabling us to code applications and implement congestion control mechanisms. T-mote Sky motes

are used to configure the border router because we use IPv6 Routing Protocol for Low-Power and Lossy Networks [19] in storing mode, which requires a larger RAM capacity provided by Sky motes. The relevant simulation parameters are summarized in Table 3.

Operating system	Contiki 3.0
Simulator	Cooja
Area	Linear – varies from 20 m * 10 m to 500 m * 20 m, Random - ranges from 50m * 50 m to 400 m * 400 m
Radio Medium	Unit Disk Graph Medium (UDGM) - Distance loss
Mote type	T-mote sky and Zolerita Z1
Number of nodes	40, 80, 120, and 160
Topologies	Linear (chain) and Random
Node transmission range	10 m
Node interference range	20 m
Interference level	10%, 25%, 35%, 50%
Traffic type	Constant bit rate (periodic)
Speed of nodes	Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr
Packet size	100 bytes
Retransmission limit	4
Simulation time	300 seconds

Table 3: Simulation parameters

Figure 6 shows the topologies used, nodeID 1 represents the RPL border router, nodeID 2 represents the CoAP server, and the rest are CoAP clients. After the RPL setup, the clients periodically send messages targeted at the servers. Every scenario is run 15 times with a different random seed with a simulation time of 300 seconds. The simulation results are plotted against the number of nodes, interference level, and speed of nodes using a box-and-whisker plot diagram. To determine the difference among the protocols, we used Tukey's Honestly Significance test with a confidence level of 90%, evaluated using R programming [21].



(a) Linear topology (b) Random topology

Figure 6: Topologies (a) Linear (b) Random

The metrics used in the performance evaluation are i) the *goodput*, measured as the total amount of data successfully received per unit of time; ii) the *packet delivery ratio*, computed as the ratio of the total number of received CoAP messages over the number of sent CoAP messages in a given time interval; iii) the *retransmission rate*, evaluated as the percentage of retransmitted packets over the total number of packets sent by the CoAP clients; and iv) the *packet sending rate* at the application level, the total number of packets sent by the CoAP clients in a given time interval.

Next, we present the aggregated results which show the overall performance of our protocols.

4.3.2 Results

Linear scenario

Goodput

Figure 7 shows the comparative goodput of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The goodput achieved in the linear scenario with ACCB is statistically higher by approx. 49.5%, 436.5%, 312.7% than CoAP, CoCoA+, and AdCoCoA (p-value = 0 < 0.10) respectively. Figure 7 (a) shows goodput against the protocols for the number of nodes 40, 80, 120, and 160. The goodput of ACCB is significantly higher by approx. 41%, 417.1%, 221.1% than CoAP, CoCoA+, and AdCoCoA respectively. As the number of nodes increases, the goodput also increases. Figure 7 (b) shows goodput against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is significantly higher by approx. 45.6%, 441.4%, 245.4% than CoAP, CoCoA+, and AdCoCoA respectively. As the interference level increases, the goodput of decreases. Figure 7 (c) shows goodput against the protocols for speed of nodes Zero km/hr, 15-

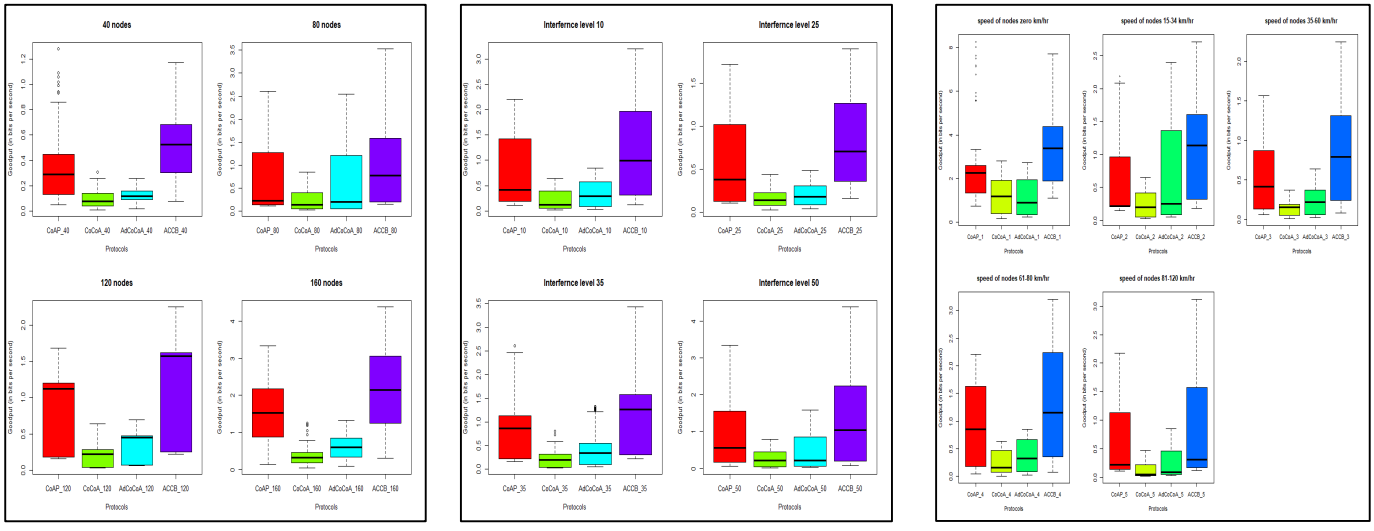
34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The goodput of ACCB is significantly higher by approx. 45.6%, 410.9%, 205.6% than CoAP, CoCoA+, and AdCoCoA respectively.

Packet delivery ratio

Figure 8 shows the comparative packet delivery ratio of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The packet delivery ratio achieved in the linear scenario with ACCB is statistically higher by approx. 10.12%, 56.05%, 28.3% than CoAP, CoCoA+, and AdCoCoA (p-value is 1.7E-11 < 0.10) respectively. Figure 8 (a) shows the packet delivery ratio against the protocols for the number of nodes 40, 80, 120, and 160. For 40 nodes, the goodput of ACCB is significantly higher by approx. 9.2%, 53.3%, 26.9% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 8 (b) shows packet delivery ratio against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is statistically higher by approx. 10.2%, 39.5%, 20.4% than CoAP, CoCoA+, and AdCoCoA as respectively. Figure 8 (c) shows packet delivery ratio against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The packet delivery ratio of ACCB is statistically higher by approx. 6.4%, 41.3%, 23.1% than CoAP, CoCoA+, and AdCoCoA as respectively.

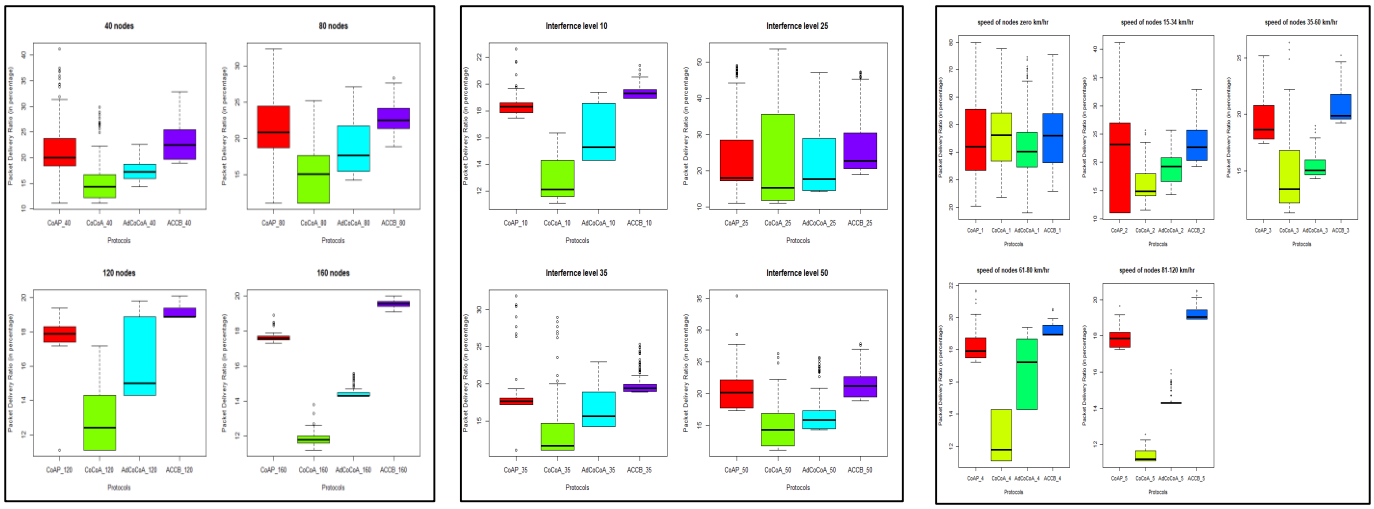
Retransmission ratio

Figure 9 shows the comparative retransmission ratio of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The retransmission rate achieved in the linear scenario with ACCB is the same as CoAP, CoCoA+, and AdCoCoA, respectively. The retransmission rate is directly proportional to the number of nodes. As the interference level increases, the retransmission rate also increases. As the speed of nodes increases, the retransmission rate also increases.



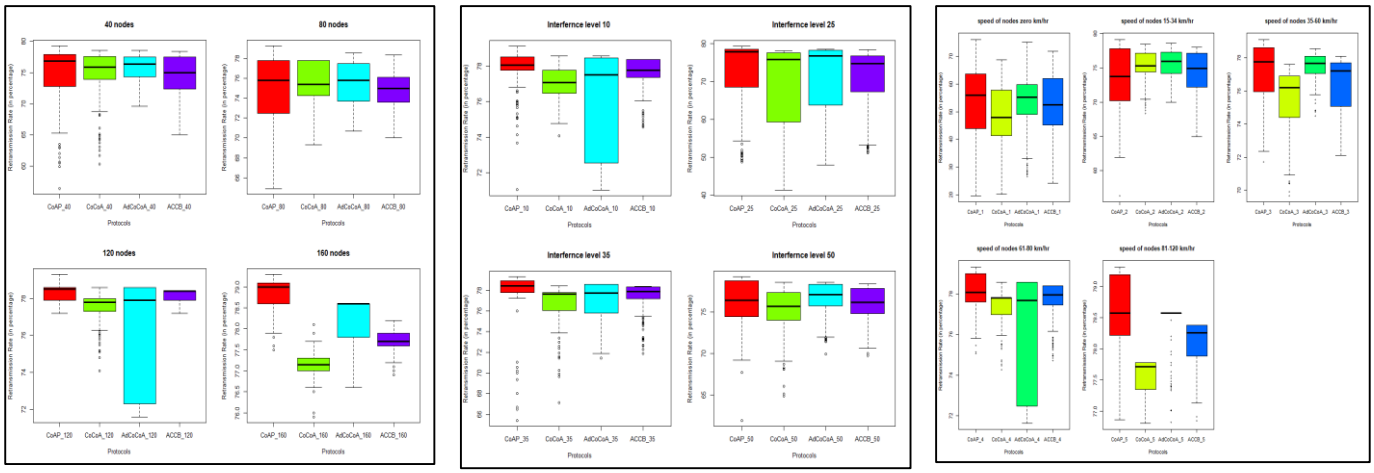
(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 7: *Goodput v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes*



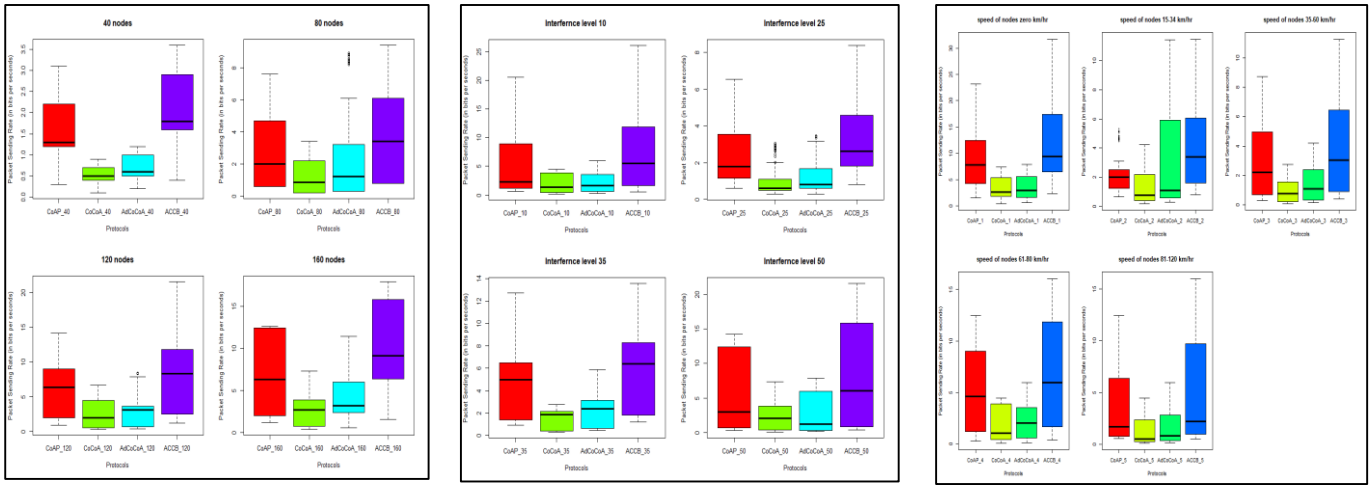
(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 8: *Packet delivery ratio v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes*



(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 9: Retransmission rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



(a)Varying number of nod *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 10: Packet sending rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying the speed of nodes

Packet sending rate

Figure 10 shows the comparative packet sending rate of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The packet sending rate achieved in the linear scenario with ACCB is statistically higher by approx. 37.7%, 265%, 175.3% than CoAP, CoCoA+, and AdCoCoA (p-value is $0 < 0.10$) respectively. Figure 10 (a) shows the packet sending rate against the protocols for the number of nodes 40, 80, 120, and 160. The packet-sending rate of ACCB is significantly higher by approx. 39.9%, 255%, 136% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 10 (b) shows packet sending rate against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The packet-sending rate of ACCB is significantly higher by approx. 37.5%, 272.3%, 180.7% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 10 (c) shows packet sending rate against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The packet-sending rate of ACCB is significantly higher by approx. 41.7%, 260.4%, 165.9% than CoAP, CoCoA+, and AdCoCoA respectively.

Random scenario:

Goodput

Figure 11 shows the comparative goodput of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The goodput achieved in the random scenario with ACCB is statistically higher by approx. 60.5%, 482%, 202.1% than CoAP, CoCoA+, and AdCoCoA (p-value is $0 < 0.10$) respectively. Figure 11 (a) shows goodput against the protocols for the number of nodes 40, 80, 120, and 160. The goodput of ACCB is significantly higher by approx. 63.8%, 482.4%, 211.7% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 11 (b) shows goodput against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is significantly higher by approx. 61.6%, 500.2%, 228% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 11 (c) shows goodput against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The goodput of ACCB is significantly higher by approx. 65.5%, 552.2%, 218% than CoAP, CoCoA+, and AdCoCoA respectively.

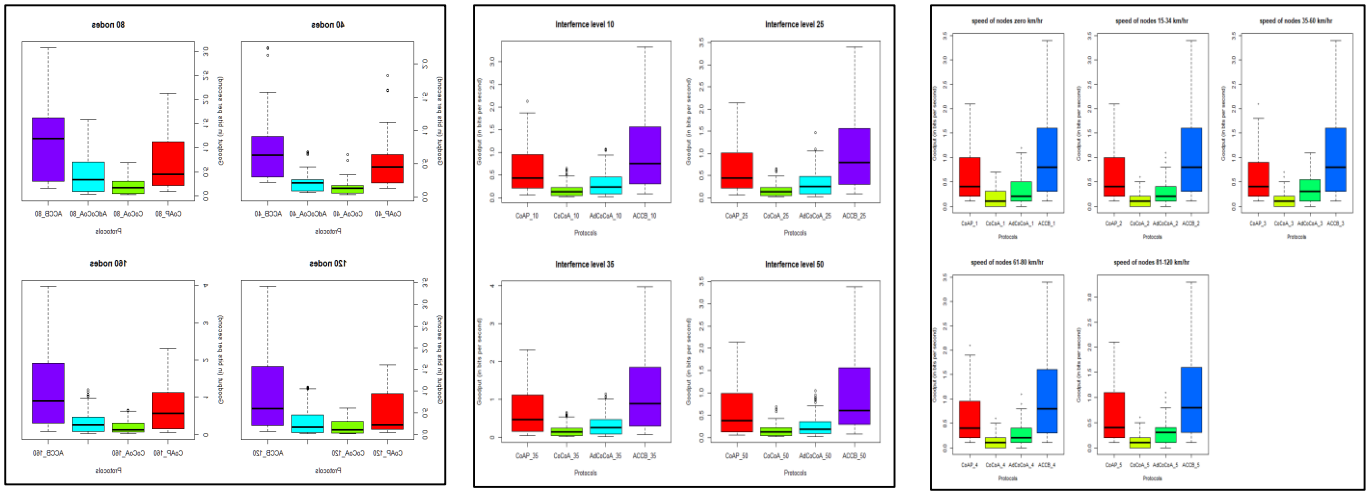
Packet Delivery Ratio

Figure 12 shows the comparative packet delivery ratio of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The packet delivery ratio achieved in the random scenario with ACCB is significantly higher by approx. 7.6%, 60.6%, 26% than CoAP, CoCoA+, and AdCoCoA (p-value is $2.5E\text{-}11 < 0.10$) respectively. Figure 12 (a) shows the packet delivery ratio against the protocols for the number of nodes 40, 80, 120, and 160. The goodput of ACCB is significantly higher by approx. 7.7%, 60.6%, 25.3% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 12 (b) shows packet delivery ratio against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is significantly higher by approx. 7.8%, 64.8%, 30% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 12 (c) shows packet delivery ratio against the protocols for speed of nodes Zero km/hr, 15- 34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The packet delivery ratio of ACCB is significantly higher by approx. 8.1%, 59%, 29.1% than CoAP, CoCoA+, and AdCoCoA respectively.

Retransmission ratio

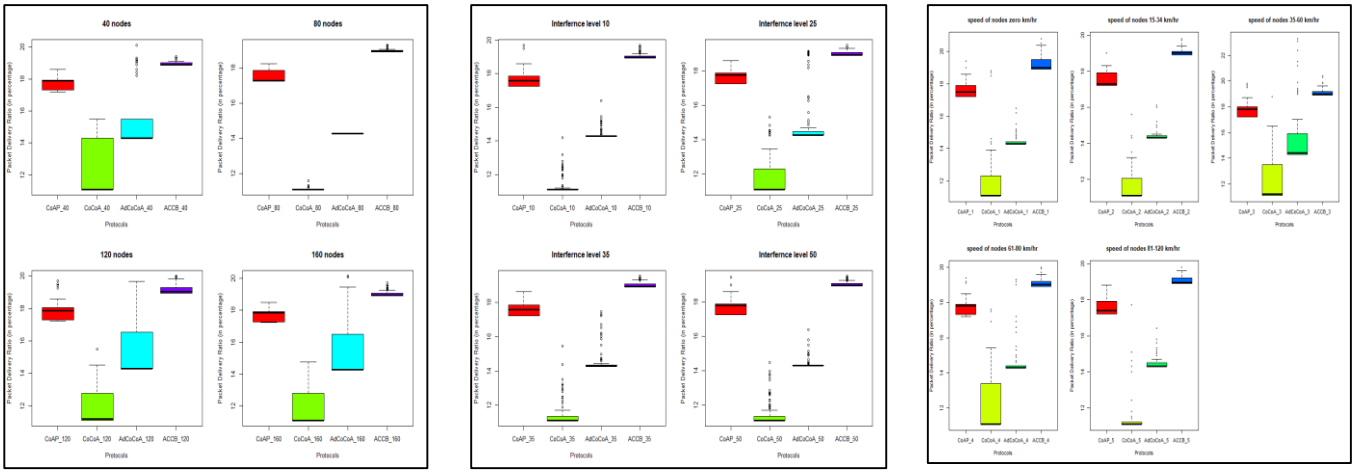
Figure 13 shows the comparative retransmission ratio of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The retransmission rate achieved in the random scenario with ACCB is the same

as CoAP, CoCoA+, and AdCoCoA, respectively. Figure 13 (a), (b), and (c) show the retransmission rate against protocols by varying the number of nodes, interference level, and speed of nodes. As the number of nodes increases, the retransmission rate also increases. As the interference level increases, the retransmission rate also increases. As the speed of nodes increases, the retransmission rate also increases.



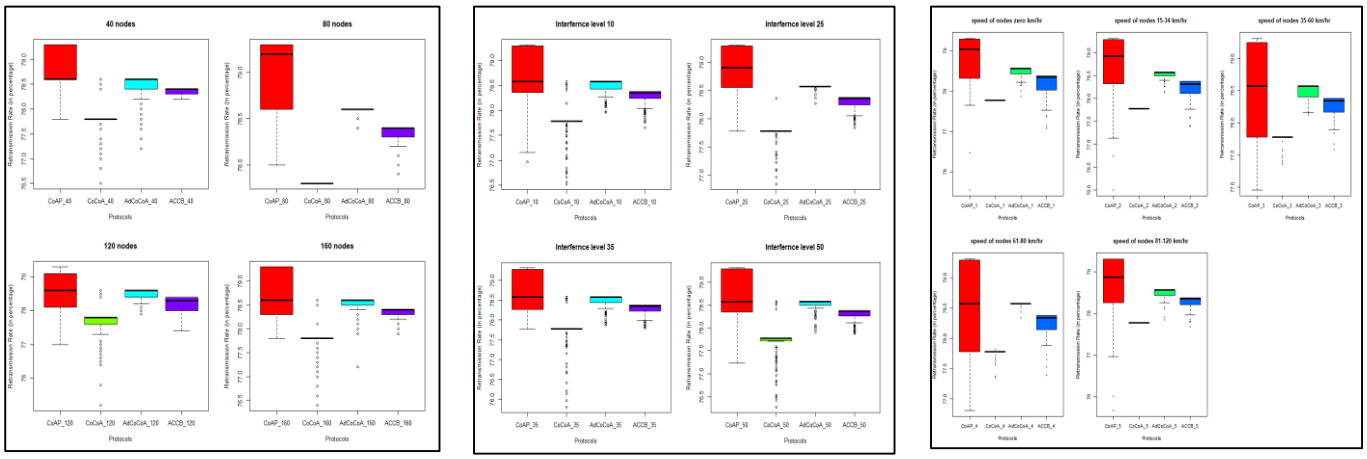
(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 11: Goodput v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



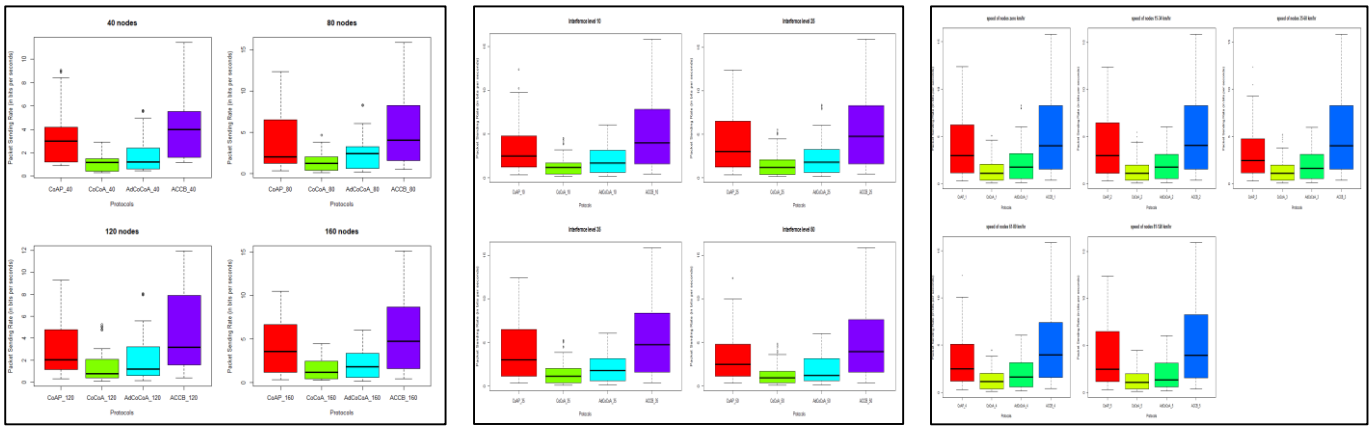
(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 12: Packet delivery ratio v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 13: Retransmission rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying speed of nodes



(a)Varying number of nodes *(b)Varying interference level* *(c) Varying speed of nodes*

Figure 14: Packet sending rate v/s protocols (a)Varying number of nodes (b)Varying interference level (c) Varying the speed of nodes

Packet sending rate

Figure 14 shows the comparative packet sending rate of CoAP, CoCoA+, AdCoCoA, and ACCB congestion control. The packet sending rate achieved in the random scenario with ACCB is statistically higher by approx. 40.9%, 284%, 146.4% than CoAP, CoCoA+, and AdCoCoA as p-value is ($0 < 0.10$) respectively. Figure 14(a) shows goodput against the protocols for the number of nodes 40, 80, 120, and 160. The goodput of ACCB is significantly higher by approx. 42.1%, 277.4%, 126.9% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 14(b) shows goodput against the protocols for interference level (IF level) 10%, 25%, 35%, 50%. The goodput of ACCB is statistically higher by approx. 41.7%, 286.1%, 150.1% than CoAP, CoCoA+, and AdCoCoA respectively. Figure 14(c) shows goodput against the protocols for speed of nodes Zero km/hr, 15-34 km/hr, 35- 60 km/hr, 61- 80 km/hr, 81- 120 km/hr. The goodput of ACCB is statistically higher by approx. 42.2%, 285.4%, 148.8% than CoAP, CoCoA+, and AdCoCoA respectively.

4.4 Discussion

Our simulation results show that ACCB performs better than CoAP, CoCoA+, and AdCoCoA in linear and random scenarios. ACCB has significantly higher goodput 49.5%, 436.5%, and 312.7% compared with CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. ACCB has significantly higher packet delivery ratio 10.1%, 56%, 23.3% compared with CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. ACCB has significantly higher packet sending rate

37.7%, 265%, and 175.3% compared with CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. The results show ACCB has significantly higher goodput (60.5%, 482%,202.1%), packet delivery ratio (7.6%, 60.6%, 26%), and packet sending rate (40.9%, 284%, 146.45%), compared to CoAP, CoCoA+ and AdCoCoA respectively in random walk scenario. ACCB has similar retransmission rate compared to CoAP, CoCoA+ and AdCoCoA in both the scenarios.

Analysis:

AdCoCoA tends to perform better because it determines dynamic scaling factors based on the change in the network. The estimated RTO is large enough; packets are received in first or minimum attempts. The dynamic factors help to increase the transmission rate and achieve higher goodput. Due to the use of dynamic scaling factors, packet loss and retransmission attempts are reduced.

ACCB estimates RTO per RTT measured and considers multiple network status parameters. The dynamic smoothing factor scales the RTO as per the change in RTT, ensuring that the maximum CoAP messages are received with lesser retransmissions. In contrast, AdCoCoA considers a single network status parameter to calculate dynamic scaling factors. CoCoA uses fixed scaling factors for calculating RTT, RTTVAR, and RTO. CoAP does not consider any network status parameter and doubles the RTO value on retransmission. AdCoCoA and ACCB ensure that the estimated RTO is large enough to ensure that packets are received with minimum retransmissions. ACCB helps increase the packet delivery ratio and sending rate, and achieves higher goodput.

5. CONCLUSION

In recent studies, most existing protocols use fixed scaling factors for estimating RTO. We have proposed multiple protocols that use dynamic scaling factors. We presented two of our protocols in this synopsis. We have proposed an Adaptive congestion control algorithm (AdCoCoA) for CoAP in IoT networks. AdCoCoA has a dynamic-scaling-factor-based mechanism for RTO estimation. AdCoCoA dynamically computes the RTO depending on the changes noticed in the network status. This helps improve the performance and minimize retransmissions. Our simulation results show that AdCoCoA performs better than CoCoA+ in different topologies. The goodput of AdCoCoA is higher by 71.3% as compared to CoCoA+. The packet delivery ratio of AdCoCoA is higher by 36.4% compared to CoCoA+. The retransmission index in the case of AdCoCoA is lower than CoCoA+ by 11.6%. The transmission rate in the case of AdCoCoA is higher by 55.6% compared to CoCoA+. AdCoCoA tends to perform better because it determines dynamic scaling factors based on the change in the network. If the estimated RTO is large enough, packets are received in the first or minimum attempts. This RTO helps to increase the transmission rate and achieve higher goodput. Due to dynamic scaling factors, packet loss and retransmission attempts are reduced.

Our second protocol, Adaptive congestion control with a backoff algorithm (ACCB), has a dynamic-scaling-factor-based mechanism for RTT, RTTVAR, and RTO estimation. ACCB adaptively computes the RTO depending on the changes in the network status. ACCB considers RTT deviation, retransmission ratio, and packet loss ratio to calculate dynamic scaling and smoothing factors. Adaptive factors and ABF mechanisms help improve the performance and minimize retransmissions.

Our simulation results show that ACCB performs better than CoAP, CoCoA+, and AdCoCoA in linear and random scenarios. ACCB has significantly higher goodput 49.5%, 436.5%, and 312.7% compared with CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. ACCB has significantly higher packet delivery ratio 10.1%, 56%, 23.3% compared with CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. ACCB has significantly higher packet sending rate 37.7%, 265%, and 175.3% compared with CoAP, CoCoA+ and AdCoCoA respectively in linear scenario. The results show ACCB has significantly higher goodput (60.5%, 482%,202.1%), packet delivery ratio (7.6%, 60.6%, 26%), and packet sending rate (40.9%, 284%, 146.45%), compared to CoAP, CoCoA+ and AdCoCoA respectively in random walk scenario. ACCB has similar retransmission ratio compared to CoAP, CoCoA+ and AdCoCoA in both the scenarios.

6. FUTURE SCOPE

Limitations

ACCB does not directly measure the status of the network. This can lead to poor results in a certain scenarios.

Future scope

ACCB can be extended into a new protocol which considers multiple inputs from the network parameters and scenarios for estimating dynamic scaling factors and backoff mechanism. For example, a protocol which adapts the backoff mechanism based on the actual speed of the node, or the interference level on the link.

A cross layer protocol can be developed which acquires information from lower layers (for example –layer2 for collision status, layer3 for route information, etc.) to estimate dynamic scaling factors and adapts the backoff mechanism.

7. PUBLICATIONS

Journals

1. " ACCB – Adaptive Congestion Control with Back-off for CoAP " accepted in International Journal of Computer Science and Network Security (IJCSNS) (Scopus indexed) (2022)
2. " A Survey on Congestion Control Protocols for CoAP " revised article submitted to International Journal of Communication Networks and information security (IJCNIS) (Scopus indexed) (2022)

Conferences

1. S. Deshmukh and V. T. Raisinghani, "AdCoCoA- Adaptive Congestion Control Algorithm for CoAP," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-7, doi: 10.1109/ICCCNT49239.2020.9225315. (Scopus indexed) - Published paper

REFERENCES

[1] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP),"RFC 7252, p. 112, 2014.

[2] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "Congestion Control in Reliable CoAP Communication,"Proc. 16th ACM Int. Conf. Model. Anal. Simul. Wirel. Mob. Syst., pp. 365–372, 2013.

[3] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoCoA+: An advanced congestion control mechanism for CoAP,"Ad Hoc Networks, vol. 33, pp. 126–139, 2015.

[4] E. Balandina, Y. Koucheryavy, and A. Gurtov, "Computing the Retransmission Timeout in CoAP,"Internet Things, Smart Spaces, Next Gener. Netw., pp. 352–362, 2013.

[5] R. Bhalerao, S. Srinivasa Subramanian, and J. Pasquale, "An Analysis and Improvement of Congestion Control in the CoAP Internet-of-Things Protocol,"IEEE Annu. Consum. Commun. Netw. Conf., 2016.

[6] J. J. Lee, K. T. Kim, and H. Y. Youn, "Enhancement of congestion control of Constrained Application Protocol/ Congestion Control/Advanced for Internet of Things environment,"Int. J. Distrib. Sens. Networks, vol. 12, no. 11, 2016.

[7] S. Bolettieri, G. Tanganelli, C. Vallati, and E. Mingozzi, "pCoCoA: A precise congestion control algorithm for CoAP,"Ad Hoc Networks, vol. 80, pp. 116–129, 2018.

[8] F. Ouakasse and S. Rakrak, "An improved adaptive CoAP congestion control algorithm,"Int. J. Online Eng., vol. 15, no. 3, pp. 96–109, 2019.

[9] E. Ancillotti, S. Bolettieri, and R. Bruno, "RTT-based congestion control for the internet of things," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2018, vol. 10866 LNCS, pp. 3–15.

[10] V. J. Rathod, S. Krishnam, A. Kumar, G. Baraskar, and M. P. Tahiliani, "Effective RTO estimation using Eifel Retransmission Timer in CoAP,"Proc. CONECCT 2020 - 6th IEEE Int. Conf. Electron. Comput. Commun. Technol., no. ii, 2020.

[11] S. Deshmukh and V. T. Raisinghani, "AdCoCoA- Adaptive Congestion Control Algorithm for CoAP,"2020 11th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2020, 2020.

[12] G. A. Akpakwu, G. P. Hancke, and A. M. Abu-Mahfouz, "CACC: Context-aware congestion control approach for lightweight CoAP/UDP-based Internet of Things traffic,"Trans. Emerg. Telecommun. Technol., vol. 31, no. 2, pp. 1–19, 2020.

[13] I. Jarvinen, I. Raitahila, Z. Cao, and M. Kojo, "FASOR Retransmission Timeout and Congestion Control Mechanism for CoAP," in 2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings, 2019.

[14] S. Bansal and D. Kumar, "Distance-based congestion control mechanism for CoAP in IoT,"IET Commun., vol. 14, no. 19, pp. 3512–3520, 2020.

[15] M. P. T. Vishal Rathod, Natasha Jeppu, Samanvita Sastry, Shruti Singala, "CoCoA++: Delay gradient based congestion control for Internet of Things,"Futur. Gener. Comput. Syst., vol. Volume 100, p. Pages 1053-1072, May 2019.

[16] H. Hellaoui and M. Koudil, "Bird Flocking Congestion Control for CoAP/RPL/6LoWPAN Networks,"IoT-Sys '15 Proc. 2015 Work. IoT challenges Mob. Ind. Syst., pp. 25–30, 2015.

[17] A. P. Castellani, M. Rossi, and M. Zorzi, "Back Pressure Congestion Control for CoAP/6LoWPAN Networks,"Ad Hoc Networks, vol. 18, pp. 71–84, 2013.

[18] E. Ancillotti, R. Bruno, C. Vallati, and E. Mingozzi, "Design and Evaluation of a Rate-Based Congestion Control Mechanism in CoAP for IoT Applications," in 19th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2018, 2018.

[19] E. Ancillotti and R. Bruno, "BDP-CoAP: Leveraging Bandwidth-Delay Product for Congestion Control in CoAP," in 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 656–66.

[20] Shelby, Zach, and Bormann, Carsten, "6LoWPAN: The Wireless Embedded Internet," in Wiley Publishing, 2010.

[21] R. Ludwig and K. Sklower, “The Eifel retransmission timer,” ACM SIGCOMM Comput. Commun. Rev., vol. 30, no. 3, pp. 17–27, 2000.

[22] R. Ludwig and R. H. Katz, "The Eifel algorithm: Making TCP robust against spurious retransmissions,"Comput. Commun. Rev., vol. 30, no. 1, pp. 30–36, 2000.

[23] V. Paxson and M. Allman, “Computing TCP’s retransmission timer (RFC6298),” Ietf, pp. 1–11, 2011.

[24] D. Bertsekas and R. Gallager, Data Networks. Prentice-Hall, 1992.

[25] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," ACM Queue, vol. 14, no. 5, pp. 50:20–50:53, October 2016.

[26] Kranenburg, R.V., 2008. The Internet of Things: A Critique of Ambient Technology and the All-Seeing Network of RFID, Institute of Network Cultures.

[27] I. J¨arvinen, M. Kojo, I. Raitahila, and Z. Cao, "Fast-Slow Retransmission and Congestion Control Algorithm for CoAP," Internet Draft, Jun. 2018, Work in progress.

[28] R. K. Yadav, N. Singh, and P. Piyush, "Genetic CoCoA++: Genetic Algorithm based Congestion Control in CoAP," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 808-813, DOI: 10.1109/ICICCS48265.2020.9121093.

[29] K.K. Jonassen, Implementing CAIA delay-gradient in Linux (Master's thesis), Department of Informatics, University of Oslo, Norway, 2015.

[30] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," ACM Queue, vol. 9, no. 11, Nov. 2011.

[31] Liberti, Leo & Lavor, Carlile & Maculan, Nelson & Mucherino, Antonio. (2012). Euclidean Distance Geometry and Applications. SIAM Review. 56. 10.1137/120875909.

[32] J. Tripathi, J. de Oliveira, and J. P. Vasseur, "A performance evaluation study of RPL: Routing protocol for low power and lossy networks," in CISS'10, March 2010, pp. 1–6.

[33] J.Katto, K.Ogura, Y.Akae, T.Fujikawa, K.Kaneko, and S.Zhou: "Simple Model Analysis and Performance Tuning of Hybrid TCP Congestion Control," IEEE Globecom 2008, Dec 2008.

[34] E. Ancillotti and R. Bruno, "Comparison of CoAP and CoCoA+ congestion control mechanisms for different IoT application scenarios," 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, 2017, pp. 1186-1192, doi: 10.1109/ISCC.2017.8024686..

[35] I. Järvinen, L. Daniel and M. Kojo, "Experimental evaluation of alternative congestion control algorithms for Constrained Application Protocol (CoAP)," 2015 IEEE 2nd World

Forum on Internet of Things (WF-IoT), Milan, 2015, pp. 453-458, doi: 10.1109/WF-IoT.2015.7389097.

[36] Bolettieri S, Vallati C, Tanganelli G, Mingozzi E. Highlighting some shortcomings of the CoCoA+ congestion control algorithm. In: Ad-hoc, Mobile, andWireless Networks: 16th International Conference onAdHocNetworks and Wireless, ADHOC-NOW2017, Messina, Italy, September 20-22, 2017, Proceedings. Cham, Switzerland: Springer International Publishing; 2017:213-220.

[37] I. J¨arvinen, I. Raitahila, Z. Cao, and M. Kojo, “Is CoAP Congestion Safe?” in Proceedings of the Applied Networking Research Workshop, ser. ANRW ’18, Montreal, QC, Canada. ACM, 2018, pp. 43–49.

[38] V. J. Rathod and M. P. Tahiliani, "Geometric Sequence Technique for Effective RTO Estimation in CoAP," 2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2020, pp. 1-6, doi: 10.1109/ANTS50601.2020.9342748.

[39] R. K. Yadav, N. Singh and P. Piyush, "Genetic CoCoA++: Genetic Algorithm based Congestion Control in CoAP," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 808-813, doi: 10.1109/ICICCS48265.2020.9121093.

[40] J. Garba and S. Sani, “mlCoCoA: a machine learning-based congestion control for CoAP,” *Turk J Elec Eng Comp Sci*, pp. 11–13, 2018.

[41] S. Gheisari and E. Tahavori, “CCCLA: A cognitive approach for congestion control in Internet of Things using a game of learning automata,” *Comput. Commun.*, vol. 147, pp. 40–49, Nov. 2019.

[42] P. Aimtongkham, P. Horkaew, and C. So-In, “An Enhanced CoAP Scheme Using Fuzzy Logic with Adaptive Timeout for IoT Congestion Control,” *IEEE Access*, vol. 9, pp. 58967–58981, 2021.

[43] D. H. Hoang, T. Thuy, and D. Le, “RCOAP : A Rate Control Scheme for Reliable Bursty Data Transfer in IoT Networks,” *IEEE Access*, vol. PP, p. 1, 2021.

[44] A. Arsalan and R. A. Rehman, "Distance-Based Scheme for Broadcast Storm Mitigation in Named Software Defined Vehicular Networks (NSDVN)," *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2019, pp. 1-4, doi: 10.1109/CCNC.2019.8651700.

[45] T. Bellache, O. Shagdar and S. Tohme, "An alternative congestion control using an enhanced contention based forwarding for vehicular networks," *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, 2017, pp. 81-87, doi: 10.1109/WONS.2017.7888763.